ALICE/97-24
Internal Note/PAT
June 27, 1997

Kalman filtering application for track recognition and reconstruction in ALICE tracking system

B.Batyunya, Yu.Belikov

JINR, Dubna, Russia

K.Šafařík

CERN, Geneve, Switzerland

Abstract

A tracking program based on Kalman filtering algorithm was developed for simultaneous track recognition and reconstruction in ALICE TPC and ITS. Recognition efficiency of this program as well as reconstruction precision were estimated on samples of Monte-Carlo simulated event.

1 Introduction

Kalman filtering is a very powerful method for statistical estimations and predictions. The conditions of the Kalman filtering applicability are as follows. Let's assume a certain "system" is determined at any time moment t_k by a state vector x_k. This system varies its state from time t_{k-1} to time t_k according to an evolution equation

$$x_k = f_k(x_{k-1}) + \epsilon_k.$$

It is supposed that f_k is a known determenistic function and ϵ_k is a random vector of intrinsic "process noise" which has a zero mean value ($< \epsilon_k >= 0$) and a known covariance matrix ($cov\epsilon_k = Q_k$). Generally, only a function of the state vector can be observed, and the result of the observation m_k is corrupted by a "measurement noise":

$$m_k = h_k(x_k) + \delta_k$$

It is also assumed that the measurement is unbiased ($< \delta_k >= 0$) and has a definite covariance matrix ($cov\delta_k = V_k$).

The theory of the Kalman filtering is described in many books. We mention here only some of resulting formulas for the case of so called an extended Kalman filtering. In this case the measurement function h_k is a certain matrix H_k.

Let's suppose that some estimates of the state vector $\tilde{x}_{k-1}$ and of the its error matrix $\tilde{C}_{k-1}= cov(\tilde{x}_{k-1} - x_{k-1})$ are known at a certain time slot t_{k-1}. Then we can extrapolate these estimates to the next time slot t_k by means of formulas

$$\begin{aligned} \tilde{x}{}_k^{k-1} &= f_k(\tilde{x}_{k-1}) \\ \tilde{C}{}_k^{k-1} &= F_k\, \tilde{C}_{k-1}\, F_k^T + Q_k, \quad F_k = \frac{\partial f_k}{\partial x_{k-1}} \end{aligned} \tag{1}$$

This is called as "prediction". The value of predicted χ^2-increment can be also calculated:

$$(\chi^2)_k^{k-1} = (r_k^{k-1})^T (R_k^{k-1})^{-1} r_k^{k-1}, \quad r_k^{k-1} = m_k - H_k\, \tilde{x}{}_k^{k-1}, \quad R_k^{k-1} = V_k + H_k\, \tilde{C}{}_k^{k-1}\, H_k^T \tag{2}$$

The number of degrees of freedom equals to the dimension of the vector m_k.

If we have the result of the state vector measurement at the moment t_k we can combine this additional information with the prediction results to improve our state vector estimation using "filtering" procedure:

$$\begin{aligned} \tilde{x}_k &= \tilde{x}{}_k^{k-1} + K_k(m_k - H_k\, \tilde{x}{}_k^{k-1}) \\ \tilde{C}_k &= \tilde{C}{}_k^{k-1} - K_k H_k\, \tilde{C}{}_k^{k-1} \end{aligned} \tag{3}$$

Here K_k is a Kalman gain matrix

$$K_k = \tilde{C}{}_k^{k-1}\, H_k^T (V_k + H_k\, \tilde{C}{}_k^{k-1}\, H_k^T)^{-1}$$

At last, the next formula gives us the value of filtered χ^2-increment:

$$\chi^2_k = (r_k)^T (R_k)^{-1} r_k, \quad r_k = m_k - H_k \, \widetilde{x}_k \, , \quad R_k = V_k - H_k \, \widetilde{C}_k \, H_k^T$$

It can be easily shown that the predicted chi-square equals to the filtered chi-square :

$$(\chi^2)^{k-1}_k = \chi^2_k \tag{4}$$

Kalman-like solving of the tracking problem is not so new (see for example [1] and references there) and its advantages as well as shortcomings are understood now. As we know this method possess the only shortcoming. One needs realistic initial approximations for the state vector and its errors matrix to start a stable filtering procedure, but sometimes it is a real problem to get good seeds for filtering. On the other hand, advantages of this method are quite attractive.

- It is a method for simultaneous track recognition and reconstruction.
- When one applies this method for track fitting only, the wrong hits (as a consequence of an unideality of the other recognition procedure) can be rejected during the only fitting pass due to the property (4), while these background hits are found only after the first fitting step, when a global fit method is used, and an additional fitting pass is necessary.
- When multiple scattering is not negligible and, therefore, track measurements are correlated, we have to manipulate with large matrices in the global fit. For example, if we have 100 hits per track, the size of the global covariance matrix is 100×100, whereas there are about 100 matrices of 5×5 dimensions for the Kalman filtering. It is clear that calculations are faster in the last case.
- The Kalman filtering is very useful to find a prolongation of a track from one detector to another. Unlike the global fit the Kalman filtering gives a *local* track estimate at the detector boundary. The *local* estimate is often better for the track prolongation as compared with the estimate *averaged* over all detector sensitive volume.

In this note we describe some more manner to convert tracking problem to the Kalman filtering. Of course, our approach doesn't differ in principal from others, but we think it is more simple, explicit and convenient for object oriented programming. We present also some results of this approach application to the simulation data, which have been obtained for the ALICE tracking [2] system containing the Time Projection Chamber (TPC) and the silicon Inner Tracking System (ITS).

The tracking procedure is described in Section 2. Some problems of the TPC-ITS matching and the tracking in the ITS are discussed in Section 3. The main results are presented in Section 4 and some formulas for multiple scattering matrix evaluation and dE/dX calculation are given in Appendix (for all these calculation we assume that the particles are pions).

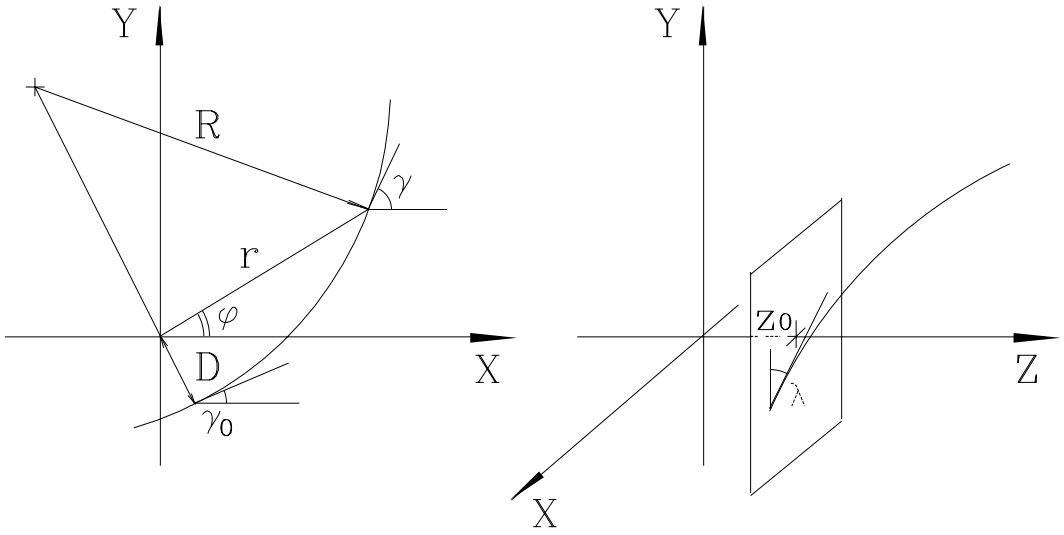

Figure 1:

2 Tracking procedure in ALICE TPC

Let us choice a track parametrization in the next form.

$$\begin{cases} \phi(r) &= \gamma_0 + \arcsin\dfrac{Cr + (1+CD)D/r}{1+2CD} \\ z(r) &= z_0 + \dfrac{\tan\lambda}{C}\arcsin(C\sqrt{\dfrac{r^2-D^2}{1+2CD}}), \end{cases}$$

where ϕ, z and r are cylindrical coordinates of a given track point, λ is a track dip angle, $C = 1/(2R)$ is a half track curvature, γ_0 is a slope of the track projection in (XY)-plane at the point of a minimal distance from the coordinate origin and D, z_0 are track transverse and longitudinal impact parameters respectively (see Fig. 1). Depending on particle charge and track position relative to the coordinate origin, parameters C and D can possess a different sign.

To construct Kalman filter we have to define the "state vector" for tracks, the propagation function f_k, matrices F_k, Q_k, H_k, V_k and so on (k-index means the k-th detector layer). With a given track parametrization we can define the track state vector as

$$x_k^T = (\phi_k, z_k, D, \tan\lambda, C).$$

Then, because the vector of measurements and its error matrix are clear

$$m_k = \begin{pmatrix} r_k\phi_k \\ z_k \end{pmatrix}, \quad V_k = \begin{pmatrix} \sigma^2_{r\phi} & 0 \\ 0 & \sigma^2_z \end{pmatrix},$$

the measurement matrix is

$$H_k = \begin{pmatrix} r_k & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

The formulas for the filtering procedure are entirely the same as for the common case (3).

It is not so hard to show that prediction for our state vector must be written in the form

$$\tilde{x}{}_k^{k-1} = f_k(\tilde{x}_{k-1}) \Longleftrightarrow \begin{cases} \overset{\sim}{\phi}{}_k^{k-1} &= \overset{\sim}{\phi}_{k-1} + \arcsin(A_k) - \arcsin(A_{k-1}) \\ \tilde{z}{}_k^{k-1} &= z_{k-1} + \dfrac{\overset{\sim}{\tan\lambda}_{k-1}}{\tilde{C}_{k-1}}\Big[\arcsin(B_k) - \arcsin(B_{k-1})\Big] \\ \tilde{D}{}_k^{k-1} &= \tilde{D}_{k-1} \\ \overset{\sim}{\tan\lambda}{}_k^{k-1} &= \overset{\sim}{\tan\lambda}_{k-1} \\ \tilde{C}{}_k^{k-1} &= \tilde{C}_{k-1}, \end{cases}$$

where $A_k = \dfrac{\tilde{C}_k\, r_k + (1+ \tilde{C}_k \tilde{D}_k)\, \tilde{D}_k\, / r_k}{1 + 2\, \tilde{C}_k \tilde{D}_k}$ and $B_k = \tilde{C}_k \sqrt{\dfrac{r_k^2 - \tilde{D}{}_k^2}{1 + 2\, \tilde{C}_k \tilde{D}_k}}$. We can use the equation (1) for the error matrix prediction with the F_k-matrix calculated from the system above

$$F_k = \left.\frac{\partial(\phi_k^{k-1}, z_k^{k-1}, D_k^{k-1}, \tan\lambda_k^{k-1}, C_k^{k-1})}{\partial(\phi_{k-1}, z_{k-1}, D_{k-1}, \tan\lambda_{k-1}, C_{k-1})}\right|_{x_{k-1}=\tilde{x}_{k-1}}.$$

At last, we have to evaluate matrix the Q_k, which describes multiple scattering distortion (appendix A.), and take into account energy losses (appendix B.).

The tracking program consists of two parts:

1. track starting part;
2. track following part.

In the track starting part we have to obtain seeds for a track and track error matrix. The well known way of track starting is an association of the closest hits to "chains" and fitting them globally to get the fit results as seeds for the Kalman filtering. This method requires an external fitting routine. It is not so convenient, because the track starting part and the track following part are quite different.

There is another way to start tracks. We start from the outermost TPC layer (labelled as no.0), get a hit with coordinates (r_0, ϕ_0, z_0) and errors $\sigma_{\phi 0}, \sigma_{z0}$ and construct a rough track and rough error matrix

$$\tilde{x}_0 = \begin{pmatrix} \phi_0 \\ z_0 \\ 0 \\ z_0/r_0 \\ 0 \end{pmatrix}, \tilde{C}_0 = \begin{pmatrix} M_\phi^2 & 0 & 0 & 0 & 0 \\ 0 & M_z^2 & 0 & 0 & 0 \\ 0 & 0 & M_D^2 & 0 & 0 \\ 0 & 0 & 0 & M_{tg\lambda}^2 & 0 \\ 0 & 0 & 0 & 0 & M_C^2 \end{pmatrix},$$

where $M_{\phi,z,D,tg\lambda,C}$ are "reasonable" large numbers. We have chosen

$$M_\phi = \sigma_\phi, \quad M_z = \sigma_z, \quad M_D = D_{min}/\sqrt{12}, \quad M_{tg\lambda} = 1/\sqrt{12}, \quad M_C = 2/(r_0\sqrt{12})$$

with $D_{min} = 10$ mm. These rough objects must be propagated to the next layer (layer no.1)

$$\tilde{x}_1^0 = f_1(\tilde{x}_0), \quad \tilde{C}_1^0 = F_1 \, \tilde{C}_0 \, F_1^T$$

Then, after propagation to the next layer, a window around current track position $(\tilde{\phi}_1^0, \tilde{z}_1^0)$ is calculated:

$$\tilde{\phi}_1^0 \pm 3\sqrt{(\tilde{C}_1^0)_{\phi\phi} + \sigma_{\phi 1}^2},$$
$$\tilde{z}_1^0 \pm 3\sqrt{(\tilde{C}_1^0)_{zz} + \sigma_{z1}^2}.$$

All hits inside the window must be considered as possible track prolongations, since the precision of the error matrix estimate $\tilde{C}_1^0$ is not enough at this step and χ^2 can not be evaluated correctly. Moreover, the filtering procedure (3) is unstable here for the same reason. But the next trick saves us. We create a temporary track $\tilde{y}\ (r_1, \phi_1, z_1)$ and its errors matrix $\tilde{E} = cov(\tilde{y} - y)$ for each hit (r_1, ϕ_1, z_1) inside our window

$$\tilde{y} = \begin{pmatrix} \phi_1 \\ z_1 \\ 0 \\ \frac{(z_0 - z_1)}{l} \\ \frac{sin(\phi_0 - \phi_1)}{l} \end{pmatrix}, \quad \tilde{E} = \begin{pmatrix} \sigma_{\phi 1}^2 & 0 & 0 & 0 & -\frac{\sigma_{\phi 1}^2}{l} \\ 0 & \sigma_{z1}^2 & 0 & -\frac{\sigma_{z1}^2}{l} & 0 \\ 0 & 0 & (10M_D)^2 & 0 & 0 \\ 0 & -\frac{\sigma_{z1}^2}{l} & 0 & 2(\frac{\sigma_{z1}}{l})^2 & 0 \\ -\frac{\sigma_{\phi 1}^2}{l} & 0 & 0 & 0 & 2(\frac{\sigma_{\phi 1}}{l})^2 \end{pmatrix},$$

where $l = \sqrt{r_0^2 + r_1^2 - 2r_0 r_1 cos(\phi_0 - \phi_1)}$. Then we can improve the current track and the error matrix estimates as follows

$$\begin{aligned} \tilde{C}_1^{improved} &= \left[(\tilde{C}_1^0)^{-1} + (\tilde{E})^{-1}\right]^{-1}, \\ \tilde{x}_1^{improved} &= \tilde{C}_1^{improved} \left[(\tilde{C}_1^0)^{-1} \, \tilde{x}_1^0 + (\tilde{E})^{-1} \, \tilde{y}\right]. \end{aligned} \tag{5}$$

These improved estimates can be extrapolated to the layer number 2 by the

$$\tilde{x}_2^1 = f_2(\tilde{x}_1^{improved}), \quad \tilde{C}_2^1 = F_2 \, \tilde{C}_1^{improved} \, F_2^T$$

and improved again with the hits, which are inside of recalculated window. As a result we obtain a new estimates $\tilde{x}_2^{improved}$ and $\tilde{C}_2^{improved}$ and repeat the procedure if necessary. We have found that for the ALICE TPC conditions the precision of the $\tilde{x}_2^{improved}$ and $\tilde{C}_2^{improved}$ is enough to pass them to track following part of the program.

When these initial approximations for the track and error matrix are obtained, the track following part begins. In this part we have to execute the next steps:

- track and track error matrix propagation to the next layer according to equations (1);
- calculation of the predicted $(\chi^2)_k^{k-1}$ value (2) for all hits inside the window $\overset{\sim}{\phi}{}_k^{k-1} \pm \Delta\phi_k$, where $\Delta\phi_k = 3\sqrt{(\tilde{C}{}_k^{k-1})_{\phi\phi} + \sigma^2_{\phi k}}$;
- filtering (3) with the "best" hit, which gives the minimal $(\chi^2)_k^{k-1}$ and this value is less than a certain value χ^2_{min}.

These steps must be repeated until the detector boundary will be archived or "best" hits will not be found at the three consecutive layers (*i.e.* there is a track kink or the current track is secondary).

If we succeed in track following up to the innermost TPC layer, we save this track and its error matrix. Further, all these tracks are sorted according to the curvature increasing and pass to the ITS tracking program (after the processing of all hits at the outermost TPC layer).

3 TPC-ITS matching and tracking in ALICE ITS

The dead zone between the TPC and the ITS is rather extensive and the track density inside the ITS is so hard, that the naive continuation of the tracking procedure described above is quite ineffective. The reason is a lot of hits which are revealed inside the $r\phi$-window at the outermost ITS layer. Some of these hits give χ^2-increments near the minimal one. In these circumstances the "best" hit (in a sense of χ^2) can be out of the current track with appreciable probability. We note, that the hit density decreases partially because we start matching from the most straight tracks and remove hits belonging to the reconstructed ones, but it doesn't eliminate a problem completely.

One way to overcome this obstacle is to follow all possible track prolongations inside the ITS. That means that for a given track from TPC, we check each path defined by our $r\phi - \chi^2$ windows, calculate total χ^2 for each path and get the path with the minimal value of the total χ^2 as a best track prolongation. This way guarantees that the most of the tracks will be followed from the innermost TPC layer to the innermost ITS layer. Unfortunately, a large part of these tracks will contain wrong hits and this will be a reason of a low reconstruction precision.

To reduce the wrong hit contamination, we have to take into account an information about vertex, which can be obtained *"a priori"* before the full track reconstruction (see Section 11.3.2 of the [2]). Vertex constraints can be putted in the operation by following

redefining of the measurement vector and its errors matrix:

$$m_k = \begin{pmatrix} r_k\phi_k \\ z_k \\ 0 \\ z_k \dfrac{C_k}{\arcsin C_k r_k} \end{pmatrix}, \text{ and } V_k = \begin{pmatrix} (r_k\sigma_{\phi k})^2 & 0 & 0 & 0 \\ 0 & \sigma_{zk}^2 & 0 & \dfrac{\sigma_{zk}^2}{r_k} \\ 0 & 0 & \sigma_{Dk}^2 & 0 \\ 0 & \dfrac{\sigma_{zk}^2}{r_k} & 0 & (\dfrac{\sigma_{zk}}{r_k})^2 + \sigma_{\tan\lambda k}^2 \end{pmatrix}$$

The measurement matrix H_k must be redefined correspondingly as:

$$H_k = \begin{pmatrix} r_k & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

The sense of these definitions is quite clear. The third component of the m_k means that our track have transverse impact parameter $D = 0$. The fourth component (dip angle tangent) is calculated by assuming that track goes through points at $z = z_k$ and $z = 0$.

Values of the σ_{Dk} and $\sigma_{\tan\lambda k}$ need to be evaluated apart. These values are determined by multiple scattering between the current layer number k and the vertex and, therefore, depend on a track curvature and (in much less extent) a track dip angle. Note, that hits in the ITS don't influence visibly on a precision of the curvature reconstruction and we can use the curvature reconstructed in the TPC for σ_{Dk} and $\sigma_{\tan\lambda k}$ calculation at the each ITS layer. To do this calculation we create a temporary track as:

$$\tilde{t} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ z_0 \dfrac{C_0}{\arcsin C_0 r_0} \\ C_0 \end{pmatrix} \quad cov(\tilde{t} - t) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{zv}^2 & 0 & \dfrac{\sigma_{zv}^2}{r_0} & 0 \\ 0 & 0 & \sigma_{Dv}^2 & 0 & 0 \\ 0 & \dfrac{\sigma_{zv}^2}{r_0} & 0 & \dfrac{\sigma_{zv}^2 + \sigma_{z0}^2}{r_0^2} & 0 \\ 0 & 0 & 0 & 0 & \sigma_{C0}^2 \end{pmatrix},$$

where z_0, C_0 are current track z-coordinate and curvature after an extrapolation from the TPC to the outermost ITS layer with radius r_0; $\sigma_{z0}^2 = (\tilde{C}{}_0^{TPC})_{zz}$ is a dispersion of the track z-coordinate at $r = r_0$ and σ_{zv} and σ_{Dv} are uncertainties of the vertex position which is obtained *a priori* (we put the $\sigma_{zv} \sim 1.0$ mm and $\sigma_{Dv} \sim 0.1$ mm). The σ_{Dk} and $\sigma_{\tan\lambda k}$ (in the expression for the V_k) can be consequently obtained during propagation of this temporary track from the vertex to the outermost ITS layer (see formula (1)) and are equal to

$$\sigma_{Dk}^2 = [cov(\tilde{t} - t)_k^{k+1}]_{DD}, \qquad \sigma_{\tan\lambda k}^2 = [cov(\tilde{t} - t)_k^{k+1}]_{\tan\lambda\tan\lambda}.$$

So, we have found that the best TPC-ITS matching and tracking inside the ITS can be carried out if we

- make the filtering procedure with the four-component vectors m_k taking into account the vertex position,

- check all possible track prolongations defined by the $r\phi$-windows and χ^2 increments which have four degrees of freedom in this case,
- get as the real track prolongation the path with the minimal value of the total χ^2 per path.

4 Results and discussion

The tracking simulation has been done for the ALICE tracking system using the standart SHAKER event generator [3] and the GEANT 3.21 package. The event was generated with a charged particle ($\pi^{\pm}$, $K^{\pm}$ and protons/antiprotons) density of $dN/dy = 8000$ in the polar angle region of $40^{\circ} \leq \theta \leq 140^{\circ}$ and at the particle momenta $p \geq 30 MeV/c$. The photons from π^0 decays were generated also. All secondary processes (inside the matter) were included and magnetic field equaled to $0.2T$ was taken into account. We used for the silicon ITS performance the detail simulation model [4] including the clustering algorithm and the coordinate reconstruction as a center of gravity of a charge distribution in a cluster. The ITS consists of six silicon cylindrical layers with the values of radii and the mean coordinate resolution presented in the Table 1.

Layer	R (cm)	Mean coordinate resolution (μm)	
		$r\phi$	z
1	3.9	15.	90.
2	7.6	15.	90.
3	14.	20.	30.
4	24.	20.	30.
5	40.	30.	860.
6	45.	30.	860.

Table 1: Parameters of the ITS layers

For the tracking simulation in the TPC we used a simple TPC geometry including a barrel (filled in a gas) divided to 75 very thin layers. The innermost barrel radius is 78 cm and a thickness of the barrel is 150 cm. The explicit GEANT coordinates of the track crossing points with the layers were smeared according to Gaussian resolution of 0.6 mm on $r\phi$ direction and 1.5 mm on z direction. The full amounts of a material was put equal to 4.4% X_0 and 3.4% X_0 for the ITS and the TPC respectively. Besides, the beam pipe with 0.2% X_0 has been included.

After recognition we separated tracks into several sets:

- **found** tracks, which consist of

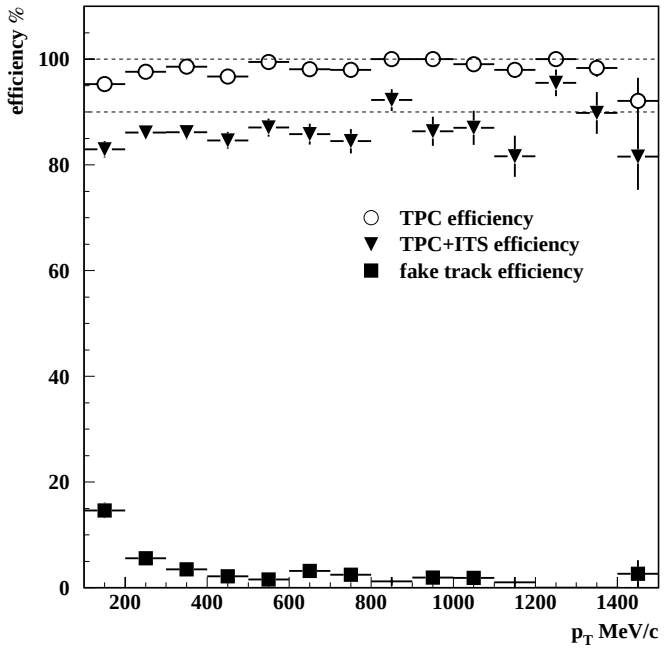

Figure 2: Tracking efficiencies

  - **good** tracks, when each track contains only right (own) measurement points in the ITS and not more than two wrong hits in the TPC,
  - and **fake** tracks otherwise;

- **initial** tracks, which are generated ones with points on the TPC boundaries and all hits inside the ITS(these tracks amount to 65% from all generated ones).

We calculated tracking efficiency as the ratio

$$\epsilon = \frac{N_{found}}{N_{initial}} = \frac{N_{good} + N_{fake}}{N_{initial}}$$

where N_{found}, N_{good}, N_{fake} and $N_{initial}$ are numbers of found, good, fake and initial tracks respectively. Figure 2 shows efficiencies for the TPC only, for both the TPC and ITS processing and fake track efficiency, which is $\epsilon_{fake} = N_{fake}/N_{initial}$.

One can see that the efficiency is quite high (near 100%) if only the TPC is used. Of course, such a high efficiency is the consequence of not so realistic TPC simulation, without the hit clustering witch one leads to overlapping of tracks and to a loss of some hits. It is seen from Fig. 2 that the efficiency decreases significantly (especially at the low p_T values) when the ITS is used. First of all, the reason is the TPC-ITS matching and multiple scattering influence. Sometimes, the hit number reaches of $10 \div 20$ inside the $r\phi - \chi^2$ windows at the outermost ITS layer. This effect manifests itself mostly in low p_T region, where the window

sizes increase because the multiple scattering. The other reason of efficiency losses is a loss of hits inside the ITS related to the track overlapping. All these effects are under study now.

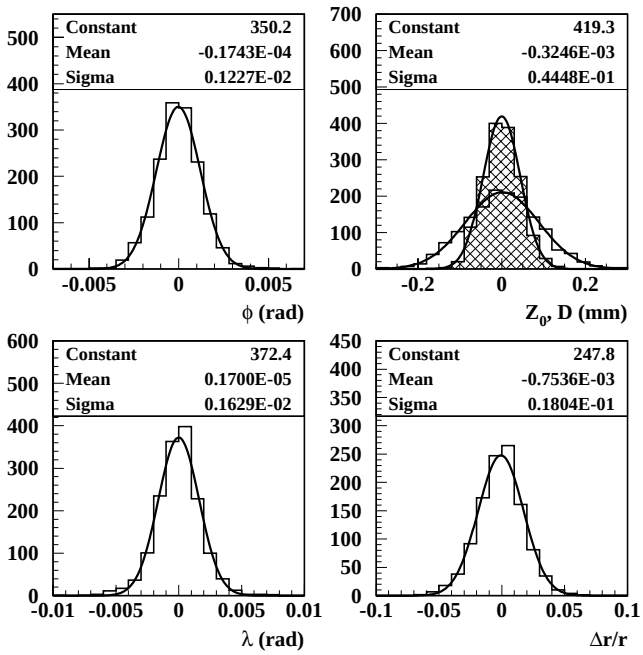

Figure 3: Track reconstruction precisions

Figure 3 shows distributions of the differenses of generated and reconstructed track parameters. It is seen that the angular resolutions (sigma) are $1.5 \div 2.0$ mrad, the momentum resolution (the same as for the $\Delta r/r$) is smaller than 2% and the impact parameter D and Z_0 resolutions are $\sim 40\mu m$ and $\sim 90\mu m$ respectively (resolutions for impact parameters were obtained by assuming that the vertex position is known with precision $\sim 50\mu m$ for Z_0 and $\sim 100\mu m$ for D). Track parameter resolutions are very near to the ones declared in the section 11.3.3 of [2], which were got from more simple tracking simulation. It should be noted that our selection of the good tracks is strong enough that one improves the resolutions but make worse the efficiency. This contradiction may be important for more real situation on account of the realistic TPC and ITS performance.

At last we give the CPU time spending amounts per a hole ALICE event processed on ION1 machine. It is about 30 min for the TPC tracking program and about 30 min for the TPC - ITS matching and ITS tracking program. Note, that this time amounts are for the both track recognition and reconstruction.

References

[1] P.Billior and S.Qian, Nucl. Instr. and Meth. A294 (1990) 219

[2] N.Ahmad et al., "ALICE Technical Proposal", CERN/LHCC/95-72, Geneve, 1995

[3] F.Antinori, Internal Note /SIM ALICE/93-09, 1993

[4] B.Batyunya, A.Zinchenko, Internal Note /SIM ALICE/94-21,1994; JINR Rapid Communications No.3[71]-95, Dubna, 1995

Appendix A. Calculation of multiple scattering matrix

A.1. *Case of discrete scatterer*

For not so thick scatterer we can assume that multiple scattering affects only to track directions leaving a track position undistorted. Using this approximation we write a general formula

$$Q_k = \frac{\partial(\phi_k, z_k, D_k, \tan\lambda_k, C_k)}{\partial(\theta_1, \theta_2)} \begin{pmatrix} <\Theta_1^2> & 0 \\ 0 & <\Theta_2^2> \end{pmatrix} \Big[\frac{\partial(\phi_k, z_k, D_k, \tan\lambda_k, C_k)}{\partial(\theta_1, \theta_2)}\Big]^T,$$

where θ_1, θ_2 are uncorrelated scattering angles in two perpendicular planes which are crossed along the momentum direction, $<\Theta_1^2>$ and $<\Theta_2^2>$ are mean squared scattering angles. Then, on account of the equalities $<\Theta_1^2>=<\Theta_2^2>\equiv<\Theta^2>=\Big[\frac{14.1}{P\beta}\Big]^2\frac{X}{X_0}$, the formula for Q_k becomes

$$\begin{aligned} Q_k &= <\Theta^2> \frac{\partial(\phi_k, z_k, D_k, \tan\lambda_k, C_k)}{\partial(\lambda_k, \gamma_k)} \frac{\partial(\lambda_k, \gamma_k)}{\partial(\theta_1, \theta_2)} \Big[\frac{\partial(\lambda_k, \gamma_k)}{\partial(\theta_1, \theta_2)}\Big]^T \Big[\frac{\partial(\phi_k, z_k, D_k, \tan\lambda_k, C_k)}{\partial(\lambda_k, \gamma_k)}\Big]^T = \\ &= <\Theta^2> J_k \begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{\cos^2\lambda_k} \end{pmatrix} J_k^T = \Big[\frac{14.1}{P\beta}\Big]^2 \frac{X_k}{X_0} S_k, \end{aligned} \tag{A.1}$$

where X_k is the scatterer thickness, $\gamma_k = \arctan\dfrac{P_y}{P_x}$ is the angle between the track XY-projection and X-axis and matrix J_k is

$$J_k = \frac{\partial(\phi_k, z_k, D_k, \tan\lambda_k, C_k)}{\partial(\lambda_k, \gamma_k)} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ \frac{\partial D_k}{\partial C_k}\frac{\partial C_k}{\partial \lambda_k} & \frac{\partial D_k}{\partial \gamma_k} \\ \frac{1}{\cos^2\lambda_k} & 0 \\ \frac{\partial C_k}{\partial \lambda_k} & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ \frac{\partial D_k}{\partial C_k} C_k \tan\lambda_k & \frac{\partial D_k}{\partial \gamma_k} \\ \frac{1}{\cos^2\lambda_k} & 0 \\ C_k \tan\lambda_k & \end{pmatrix},$$

because $\dfrac{\partial C_k}{\partial \lambda_k} = \dfrac{\partial C_k}{\partial P_T}\dfrac{\partial P_T}{\partial \lambda_k} = \dfrac{const}{P_T^2} P \sin\lambda_k = C_k \dfrac{P_L}{P_T} = C_k \tan\lambda_k.$

Partial derivatives $\frac{\partial D_k}{\partial C_k}$ and $\frac{\partial D_k}{\partial \gamma_k}$ are calculated from the expression for D_k

$$D_k = \left[\begin{array}{l} +\sqrt{R^2 + r_k^2 + 2Rr_k \sin(\phi_k - \gamma_k)} - R, \text{ for } R = \frac{1}{2C_k} \geq 0 \\ -\sqrt{R^2 + r_k^2 + 2Rr_k \sin(\phi_k - \gamma_k)} - R, \text{ for } R = \frac{1}{2C_k} < 0 \end{array} \right.$$

$$\frac{\partial D_k}{\partial C_k} = \frac{r_k^2 - D_k^2}{1 + 2C_k D_k}$$

$$\frac{\partial D_k}{\partial \gamma_k} = \pm \frac{\sqrt{r_k^2 - (C_k D_k^2 + D_k - C_k r_k^2)^2}}{1 + 2C_k D_k}$$

A.2. *Case of continuous scatterer*

For the case of infinitely thin scatterer we define infinitely small matrix dQ as (see (A.1))

$$\{dQ\}_{ij} = \Big[\frac{14.1}{P\beta}\Big]^2 \frac{dX}{X_0} \{S\}_{ij}.$$

Therefore, multiple scattering covariance matrix for the case of continuous matter between radii r_{k-1} and r_k is

$$\{Q_k\}_{ij} = \Big[\frac{14.1}{P\beta}\Big]^2 \int \{S\}_{ij} \frac{dX}{X_0} = \Big[\frac{14.1}{P\beta}\Big]^2 \int_{r_{k-1}}^{r_k} \{S(r)\}_{ij} \frac{d(X/X_0)}{dr} dr. \tag{A.2}$$

Appendix B. Energy losses treatment

For a taking into account energy losses we use a simplified Bethe - Bloch formula reduced to the case of pions:

$$\Delta E = \frac{0.153}{\beta^2} \left(\ln \frac{5940\beta^2}{1 - \beta^2} - \beta^2 \right) \Delta X. \tag{B.1}$$

With this formula we can modify a current track curvature every time, when track crosses a next detector layer:

$$C_{new} = C_{old} \left(1 - \frac{\Delta C_{old}}{C_{old}} \right) = C_{old} \left(1 - \frac{E}{p^2} \Delta E \right),$$

where C_{new} and C_{old} are track curvatures before and after a layer crossing, E and p are particle energy and momentum (which are clear functions of C_{old} and track dip angle), and ΔE is calculated from the equation (B.1).